Basic conditional process algebra

Eindhoven University of Technology
Department of Mathematics and Computing Science

Basic Conditional Process Algebra

by

J. Blanco and A. van Deursen

97/03

ISSN 0926-4515

Reports are available at:
http://www.win.tue.nl/win/cs

Computing Science Reports 97/03
Eindhoven, February 1997

Basic Conditional Process Algebra

Javier Blanco

Department of Computing Science, Formal Methods Group

TU Eindhoven, 5600 MB P.O. Box 513, The Netherlands

javier@win.tue.nl

Arie van Deursen

Department of Software Technology, CWI

Kruislaan 413, 1098 SJ Amsterdam, The Netherlands

arie@cwi.nl

Abstract

Every π-calculus expression can be translated to a term in "normal form" built from $+$, input and output prefix, match, and inaction. Many difficulties of the π-calculus are easier to understand and address at this simpler normal form level. We introduce a theory called *Basic Conditional Process Algebra* (BCPA), which we use to study these issues. BCPA is BPA extended with a conditional construct over Boolean expressions which can contain free variables. In this article, we consider a restricted setting without bound variables, since it already presents many non-trivial problems.

Note: Supported by NWO, the Netherlands Organization for Scientific Research, project 612-16-433, HOOP: Higher-Order and Object-Oriented Processes.

1991 Mathematics Subject Classification: 68Q60, 68Q10, 68Q40

1991 CR Categories: F.1.2, D.3.1, F.3.1, D.1.3.

Additional keywords and phrases: Process algebra, π-calculus, value passing, term rewriting, ACP.

1 Introduction

We consider algebras in which names can be transmitted as data. Different semantics were considered already in the original work [MPW92] according to the moment of instantiation of the variables used to transmit these names. These different semantic ideas were mainly based in two different notions of transition relation, called early and late transitions. In the early case, the variable is instantiated in the transition itself. This can be expressed (in the notation of the π-calculus) as follows:

$$x(y).P \xrightarrow{xz} P\{y := z\}$$

where z is any name not free in P. The associated semantics is defined by using the traditional (ground) bisimulation. The late semantics has a simpler transition rule

$$x(y).P \xrightarrow{x(y)} P$$

and the different instantiations of the variable y are considered in the definition of the corresponding bisimulation: a relation S is a late bisimulation if whenever PSQ and $P \xrightarrow{x(y)} P'$ then there exists a process Q' such that $Q \xrightarrow{x(y)} Q'$ and for any name z not free in P' it holds that $P\{y := z\}SQ\{y := z\}$.

However, a result from [MPW91] introduces a bisimulation that applied to two late transition systems will relate two terms if and only if they are early bisimilar. This relation inverts the quantifiers in the definition of the late bisimulation, saying that for any name there exists a (different) process which will match the transition.

It is shown that late bisimulation is strictly finer that early bisimulation for the π-calculus. It is almost immediate to see that late bisimulation is finer than the early one. To show that they are different the matching operator has to be used, due to the fact that in the π-calculus there is no syntactic construct for the free input.

A still finer equivalence was introduced in [San93] under the name of open bismulation. This equivalence has the advantage of being a congruence with respect to all the operations of the π-calculus. The variables are instantiated even later than in the late bisimulation. This fact is reflected in the way the matchings are evaluated, what shows an intrinsic difference with both early and late bisimimulations.

In [PS93] some axiomatizations of early/late equivalence and early/late congruence were introduced. The axiomatizations of the equivalences have weaker axioms for input prefixes (two process of the form $a(x) \cdot P$ and $a(x) \cdot Q$ will be equal only if $P\{X := y\} = \{X := y\}Q$ for any name y, whereas for the congruence one (obviously) has a general rule of the form $P = Q \Rightarrow aP = aQ$ for any atomic action a. Such a rule will be always implicitly assumed in our theories. The counterpart of this simple treatment of congruence is that the matching cannot be eliminated. Hence, in [PS93] some axioms are introduced to deal with it. In our algebraic framework we are only interested in congruences, since we want to use all the power of equational reasoning. The differences between the congruence and the equivalence present in [PS93] appear in our work mainly through the presence of two different predicates for equality: one like the one in [MPW92], which is true if and only if the two names are the same, and another closer to the one in [San93] which can introduce non-standard booleans (boolean values that are not equal to true or false).

1.1 Notes

No binding mechanisms needed.

Related work: Ponse and Groote [GP94] who have δ false, ε as true, negation, and a Boolean algebra which only includes true and false (in any case: either ϕ or $\neg\phi$ is true).

symbolic bisimulations [HL95], open bisimulations [San93],

2 Syntax and Semantics of BCPA

2.1 Operators of BCPA

We start with the simplest possible version of BCPA, which is BPA extended with true, false, and an if-then-else.

DEFINITION 2.1 The signature of BCPA consists of the sort P for processes:

- $P ::= A \mid \delta \mid P + P \mid P \cdot P \mid B :\rightarrow P$
- A is a set of *atoms*.
- $B ::= \text{true} \mid \text{false} \mid B \wedge B \mid B \vee B \mid \neg B \mid B \Rightarrow B \mid B \Leftrightarrow B$.
- The operator precedence is: $+ \ < \ :\rightarrow \ < \cdot$.

We write $\Sigma(\text{BCPA})$ to denote the signature of BCPA.

Both A and B are not yet filled in completely. We left them open at this moment.

2.2 Axiomatization

The axiomatization given below consist of all equations of BPA [BV95], extended with two straightforward equations for the conditional [BB94].

SPECIFICATION 2.2

$$
\begin{array}{rcll}
x + y & = & y + x & \text{[A1]} \\
(x + y) + z & = & x + (y + z) & \text{[A2]} \\
x + x & = & x & \text{[A3]} \\
(x + y) \cdot z & = & x \cdot z + y \cdot z & \text{[A4]} \\
(x \cdot y) \cdot z & = & x \cdot (y \cdot z) & \text{[A5]} \\
x + \delta & = & x & \text{[A6]} \\
\delta \cdot x & = & \delta & \text{[A7]} \\
 & & & \\
\text{true} :\rightarrow x & = & x & \text{[GC1]} \\
\text{false} :\rightarrow x & = & \delta & \text{[GC2]}
\end{array}
$$

We also need to axiomatize the booleans. However, there exist many complete axiomatizations of the booleans, and we assume that we can use any of them to prove equivalence of booleans. We only need the following conditional axiom that relates equivalence of booleans and conditionals:

SPECIFICATION 2.3

$$\beta \Leftrightarrow \gamma \Longrightarrow \beta :\to x \quad = \quad \gamma :\to x \quad [\mathrm{B}]$$

Using these equations, we can translate closed BCPA terms to *basic* terms, which only have action prefixing, not full sequential composition:

DEFINITION 2.4 Basic terms are given by the syntax:

$$T ::= \delta \mid T + T \mid B :\to A \mid B :\to A \cdot T$$

PROPOSITION 2.5 Let s be a closed BCPA term. Then there is a basic term t such that $E(\mathrm{BCPA}) \vdash t = s$.

2.3 Variables

Next, we also consider equality tests in which variables may occur. First we define an equivalence relation over Booleans:

DEFINITION 2.6 A predicate $[\_ = \_]$ over some set D is an equivalence if, for every $d, d_1, d_2, d_3 \in D$, it is:

- Reflexive: $[d = d]$;
- Symmetric: $[d_1 = d_2] \Leftrightarrow [d_2 = d_1]$;
- Transitive: $([d_1 = d_2] \wedge [d_2 = d_3] \Rightarrow [d_1 = d_3])$.

DEFINITION 2.7

- The set $\mathrm{Var} = \{\mathtt{v}, \mathtt{v0}, \mathtt{v1}, \ldots\}$ consist of an infinite number of data variables. We use (meta)variables v, w to denote elements from Var.
- The operation $[v =_{\mathsf{Var}} w]$ is the test for variable equality: it takes two variables v, w and produces a Boolean result.
- $[v =_{\mathsf{Var}} w]$ is an equivalence relation.

DEFINITION 2.8 The theory $\mathrm{BCPA}_{\mathsf{Var}}$ consists of BCPA where the Booleans B are extended with $[v =_{\mathsf{Var}} w]$.

A canonical semantics for equality of variables can be defined as follows:

DEFINITION 2.9

We define the notion of satisfaction of a boolean condition by an equivalence relation inductively on the structure of the condition

- $S \models [x = y]$ if and only if $(x, y) \in S$
- $S \models \neg\beta$ if and only if $S \not\models \beta$
- $S \models \beta \wedge \gamma$ if and only if $S \models \beta$ and $S \models \gamma$

DEFINITION 2.10 We define the notion of satisfaction of a boolean condition by a substitution inductively on the structure of the condition

- $\sigma \models [x = y]$ if and only if $\sigma(x) = \sigma(y)$
- $\sigma \models \neg\beta$ if and only if $\sigma \not\models \beta$
- $\sigma \models \beta \wedge \gamma$ if and only if $\sigma \models \beta$ and $\sigma \models \gamma$

LEMMA 2.11 Given a condition β, a substitution σ satisfies β, if and only if the (equivalence) relation S defined as

$$(x, y) \in S \Leftrightarrow \sigma(x) = \sigma(y)$$

satisfies β.

DEFINITION 2.12 Actions can be parameterized by variables, giving rise to actions $a(v_1, \ldots, v_n)$.

At this stage, actions do *not* have a binding effect; this will be discussed in a forthcoming paper.

Observe that we genuinely extended the Boolean values: in addition to true and false, we have non-standard elements like $[\mathtt{v} =_{\mathrm{Var}} \mathtt{w}]$ which are neither equal to true nor to false. This complicates the theory of $\mathrm{BCPA}_{\mathrm{Var}}$, since now the conditional cannot be eliminated anymore. For that reason, we need to extend the axiomatization of the conditional:

SPECIFICATION 2.13

$$\begin{array}{rcll} \beta :\rightarrow (x + y) & = & \beta :\rightarrow x + \beta :\rightarrow y & [\mathrm{C3}] \\ \beta :\rightarrow \gamma :\rightarrow x & = & \beta \wedge \gamma :\rightarrow x & [\mathrm{C4}] \\ \beta :\rightarrow x \cdot y & = & (\beta :\rightarrow x) \cdot y & [\mathrm{C5}] \\ \beta :\rightarrow x \cdot y & = & \beta :\rightarrow x \cdot \beta :\rightarrow y & [\mathrm{C6}] \\ x + \beta :\rightarrow x & = & x & [\mathrm{C7}] \end{array}$$

REMARK 2.14 In the literature, Baeten and Bergstra [BB92, BB95] give equations [C3], [C4] and [C5] as GC10, GC12, GC13. Equation [C6] is not sound in their system (where actions can change an environment and hence the equality between variables). They formulate [C7] in a stronger form:

$$\beta \vee \gamma :\rightarrow x = \beta :\rightarrow x + \gamma :\rightarrow x$$

from which [C7] easily follows. However, this axiom is not sound for our conditional bisimulation (see section4).

The formulation of [C7] comes from [San93, Axiom **M4**]. Sangiorgi has included [C3], and he does not need, in the more restricted syntax of the π-calculus, [C4] or [C5]. He gives [C6] as

$$[x = y]P = [x = y](P\{x/y\})$$

REMARK 2.15 We can easily see that:

- [A6] can now be derived: $x \overset{[C7]}{=} x + \text{false} :\rightarrow x \overset{[GC2]}{=} x + \delta$
- [A3] can now be derived: $x \overset{[C7]}{=} x + \text{true} :\rightarrow x \overset{[GC1]}{=} x + x$
- We have $\beta :\rightarrow \delta = \delta$: $\delta \overset{[C7]}{=} \delta + \beta :\rightarrow \delta \overset{[A6]}{=} \beta :\rightarrow \delta$
- Assume $\beta \Rightarrow \gamma$: Then $\beta :\rightarrow x \cdot \gamma :\rightarrow z = \beta :\rightarrow x \cdot z$
  Since $\beta :\rightarrow x \cdot \gamma :\rightarrow y \overset{[C6]}{=} \beta :\rightarrow x \cdot \beta :\rightarrow \gamma :\rightarrow y \overset{[C5]}{=} \beta :\rightarrow x \cdot \beta \wedge \gamma :\rightarrow y \overset{\beta \Rightarrow \gamma}{=}$ $\beta :\rightarrow x \cdot \beta :\rightarrow y \overset{[C6]}{=} \beta :\rightarrow x \cdot y$

2.4 Conditional Transition Graphs

Next, we define transition graphs where the labels are pairs consisting of a Boolean expression and the action performed.

DEFINITION 2.16 A *labeled transition system* is a 5-tuple $\langle S, L, \longrightarrow, \longrightarrow \surd, s_0 \rangle$ where

- S is a set of *states*,
- L is a set of *labels*,
- $\longrightarrow \subseteq S \times L \times S$ is a *transition relation*,
- $\longrightarrow \surd \subseteq S \times L$ is a *terminating transition relation*, and
- $s_0 \in S$ is the *initial state*.

DEFINITION 2.17 In a *conditional* labeled transition system each label $l \in L$ is a pair (β, a) with β a Boolean, and a a label from some other set.

2.5 Inference Rules for BCPA

We can produce a conditional labeled transition system from a term over BCPA by the following inference rules. The set of labels $L = B \times A$, i.e., consists of pairs of Booleans and atomic actions.

SPECIFICATION 2.18 The inference rules for the relations $\longrightarrow$ and $\longrightarrow \surd$ over BCPA are the following:

$$\text{ACT}\quad \frac{-}{a \xrightarrow{(\text{true},a)} \surd}$$

$$\text{COND}\quad \frac{x \xrightarrow{(\beta,a)} x'}{\gamma :\rightarrow x \xrightarrow{(\gamma\wedge\beta,a)} x'} \qquad \text{TCOND}\quad \frac{x \xrightarrow{(\beta,a)} \surd}{\gamma :\rightarrow x \xrightarrow{(\gamma\wedge\beta,a)} \surd}$$

$$\text{SUM-L}\quad \frac{x \xrightarrow{(\beta,a)} x'}{x + y \xrightarrow{(\beta,a)} x'} \qquad \text{SUM-R}\quad \frac{y \xrightarrow{(\beta,a)} y'}{x + y \xrightarrow{(\beta,a)} y'}$$

$$\text{TSUM-L}\quad \frac{x \xrightarrow{(\beta,a)} \surd}{x + y \xrightarrow{(\beta,a)} \surd} \qquad \text{TSUM-R}\quad \frac{y \xrightarrow{(\beta,a)} \surd}{x + y \xrightarrow{(\beta,a)} \surd}$$

$$\text{SEQ}\quad \frac{x \xrightarrow{(\beta,a)} x'}{x \cdot y \xrightarrow{(\beta,a)} x' \cdot y} \qquad \text{TSEQ}\quad \frac{x \xrightarrow{(\beta,a)} \surd}{x \cdot y \xrightarrow{(\beta,a)} y}$$

The rules indicate which transitions can be performed, and how processes affect the Boolean values associated with transitions. Rule ACT states that an atomic action can directly perform, producing the non-restrictive true value in the label condition. Rule COND (and TCOND) expresses that a conditional $\gamma :\rightarrow x$ moves its condition γ into the label, conjuncting it with the deeper condition β. The remaining rules do not interpret or change the conditional labels.

EXAMPLES 2.19

- A process $[\mathtt{v} =_{\text{Var}} \mathtt{v}] :\rightarrow a \cdot X$ can do one (true, a) step.
- A process $[\mathtt{v} =_{\text{Var}} \mathtt{w}] :\rightarrow a \cdot X$ can do one $([\mathtt{v} =_{\text{Var}} \mathtt{w}], a)$ step.

2.6 Unconditional Inference Rules

An unconditional transition system can be obtained for the syntax of BCPA in the following way. Use actions as labels, and use the conditional transition system, ignoring the Boolean elements of the labels. Furthermore, replace rule COND by:

$$\text{UNCOND} \quad \frac{x \xrightarrow{a} s}{\text{true} :\rightarrow x \xrightarrow{a} s}$$

The difference is that in the conditional case, we can continue if we have a Boolean expression of which we do not know the value; with the unconditional inference rules we have to block as long as we are not sure that it is equal to true.

2.7 Conditional Bisimulation

We define *conditional* bisimulation over conditional transition systems. This bisimulation ignores actions for which the label has become false. Moreover, if a process is to simulate another, its conditions should be at least as strong as the ones from the process it is simulating.

DEFINITION 2.20 Let $T = \langle S, L, \longrightarrow, \longrightarrow \surd, s_0 \rangle$ be conditional transition system. A relation $\mathcal{R} \subseteq S \times S$ is a *simulation* if $x\mathcal{R}y$ implies

- Whenever $x \xrightarrow{(\beta,a)} x'$, with $\beta \neq \text{false}$, there exist y', γ such that:
  1. $y \xrightarrow{(\gamma,b)} y'$, with
  2. $\beta :\rightarrow a = \beta :\rightarrow b$,
  3. $\beta \Rightarrow b$, and
  4. $(\beta :\rightarrow y')\mathcal{R}(\beta :\rightarrow y')$.
- Whenever $u \xrightarrow{(\beta,a)} \surd$, with $\beta \neq \text{false}$, there exist γ such that:
  1. $v \xrightarrow{(\gamma,b)} \surd$, with
  2. $\beta \Rightarrow \gamma$, and
  3. $\beta :\rightarrow a = \beta :\rightarrow b$.

A relation $\mathcal{R}$ is a *bisimulation* if both $\mathcal{R}$ and its inverse $\mathcal{R}^{-1}$ are simulations. We write $x \leftrightarrows y$ if there is a bisimulation $\mathcal{R}$ with $x\mathcal{R}y$.

2.8 Completeness

PROPOSITION 2.21 Bisimulation equivalence for BCPA is a congruence.

PROPOSITION 2.22 (Soundness:) If $E(\text{BCPA}) \vdash x = y$ then $x \leftrightarrows y$.

LEMMA 2.23

1. $x \xrightarrow{(\beta,a)} \surd \Longrightarrow E(\text{BCPA}) \vdash x = x + \beta :\rightarrow a,$
2. $x \xrightarrow{(\beta,a)} x' \Longrightarrow E(\text{BCPA}) \vdash x = x + \beta :\rightarrow a \cdot x'.$

PROPOSITION 2.24 (Completeness:) If $x \leftrightarrows y$ then $E(\mathrm{BCPA}) \vdash x = y$.

PROOF Given a term x let $n(x)$ denote the number of occurrences of operators $+$ and $\cdot$ in it.

The required completeness result is a consequence of the following property that we will prove by induction on $n(x) + n(y)$, where x and y are basic terms:

$$x + y \leftrightarrows y \Longrightarrow E(\mathrm{BCPA}) \vdash x + y = y$$

We consider only the most difficult case when

$$x = \beta :\rightarrow ax'$$

by definition of the action rules we know that

$$x + y \xrightarrow{(\beta,a)} x'$$

and then, since from the premises $x + y \leftrightarrows y$ we can conclude that

$$y \xrightarrow{(\gamma,b)} y'$$

where

1. $\beta \Rightarrow \gamma$,
2. $\beta :\rightarrow a = \beta :\rightarrow b$,
3. $\beta :\rightarrow x' \leftrightarrows \beta :\rightarrow y'$

Since $\leftrightarrows$ is a congruence we can infer from 3:

$$\beta :\rightarrow x' + \beta :\rightarrow y' \leftrightarrows \beta :\rightarrow y'$$

and

$$\beta :\rightarrow x' + \beta :\rightarrow y' \leftrightarrows \beta :\rightarrow x'$$

It follows then by induction hypothesis that

$$\beta :\rightarrow x' = \beta :\rightarrow y'$$

Moreover, combined with 2. above we have

$$(\beta :\rightarrow a) \cdot \beta :\rightarrow x' = (\beta :\rightarrow b) \cdot \beta :\rightarrow y'$$

or equivalently (by axiom C6)

$$\beta :\rightarrow ax' = \beta :\rightarrow by'$$

Hence, we can prove that

$$\begin{aligned} y &\overset{[C7]}{=} y + \beta :\rightarrow y \\ &\overset{(2.23)}{=} y + \beta :\rightarrow (y + \gamma :\rightarrow by') \\ &\overset{[C3]}{=} y + \beta :\rightarrow y + \beta :\rightarrow (\gamma :\rightarrow by') \\ &\overset{[C4]}{=} y + \beta \wedge \gamma :\rightarrow by' \\ &\overset{\beta \Rightarrow \gamma}{=} y + \beta :\rightarrow by' \\ &\overset{IH}{=} y + \beta :\rightarrow ax' \\ &= y + x \end{aligned}$$

2.9 Names

The decision to have an incomplete ("open") equality predicate over variables does not correspond with the original name equality available in the π-calculus [MPW92].

DEFINITION 2.25

- The operation $[\mathtt{v} =_{\text{Name}} \mathtt{w}]$ is the test for name equality of variables: it takes two variable names $\mathtt{v}$, $\mathtt{w}$ and produces a Boolean result.
- We require that $[\mathtt{v} =_{\text{Name}} \mathtt{w}] = \text{true}$ if $\mathtt{v} = \mathtt{w}$, and that it is false otherwise.

EXAMPLE 2.26 In the π-calculus under *late* and *early* bisimulation, name equality $[x =_{\text{Name}} y]$ is taken as the semantics of the match operator $[x = y]P$.

In the π-calculus under *open* bisimulation, variable equality $[x =_{\text{Var}} y]$ is used. (see also Section 3).

Following the tradition of ACP-like algebras, we are mainly interested in congruences, which implies that we can use all the power of equational reasoning in our systems. We allow also the $[x =_{\text{Name}} y]$ predicate but change the (axiomatic) definition of substitution (see next section) In the axiomatizations of name-passing calculi introduced in [PS93] the $[x =_{\text{Name}} y]$ predicate is implicitly used when the equivalences are axiomatized but $[x =_{\text{Var}} y]$ is used instead when the respective congruences are considered.

3 Substitution

3.1 Inductive Definition

A substitution is a replacement of a variable by another. We define it equationally as follows

SPECIFICATION 3.1 We introduce four substitution operators:

- $P\{\text{Var} := \text{Var}\} \to P$ for replacement in processes;
- $B\{\text{Var} := \text{Var}\} \to B$ for Booleans; and

$$
\begin{array}{rcll}
\delta\{u := v\} & = & \delta & [\text{S1}] \\
(x + y)\{u := v\} & = & x\{u := v\} + y\{u := v\} & [\text{S2}] \\
a(v_1, \ldots v_n)\{u := v\} & = & a(v_1\{u := v\}, \ldots, v_n\{u := v\}) & [\text{S3}] \\
(a \cdot x)\{u := v\} & = & a\{u := v\} \cdot x\{u := v\} & [\text{S4}] \\
(\phi :\rightarrow x) & = & \phi\{u := v\} :\rightarrow x\{u := v\} & [\text{S5}]
\end{array}
$$

$$
\begin{array}{rcll}
\text{true}\{u := v\} & = & \text{true} & [\text{S6}] \\
\text{false}\{u := v\} & = & \text{false} & [\text{S7}] \\
(\phi \wedge \psi)\{u := v\} & = & \phi\{u := v\} \wedge \psi\{u := v\} & [\text{S8}] \\
[v_1 =_{\text{Var}} v_2]\{u := v\} & = & [v_1\{u := v\} =_{\text{Var}} v_2\{u := v\}] & [\text{S9}]
\end{array}
$$

$$
\begin{array}{rcll}
u\{u := v\} & = & v & [\text{S10}] \\
u \not\equiv u' \Rightarrow u\{u' := v\} & = & u & [\text{S11}]
\end{array}
$$

We use $x\sigma$ for a sequence $x\{u_1 := v_1\} \cdots \{u_n := v_n\}$ of substitutions.

REMARK 3.2 Adding the equation

$$[v_1 =_{\text{Name}} v_2]\{u := v\} = [v_1\{u := v\} =_{\text{Name}} v_2\{u := v\}]$$

makes the Booleans inconsistent:

$$\text{false} = \text{false}\{\mathtt{v} := \mathtt{w}\} = [\mathtt{v} =_{\text{Name}} \mathtt{w}]\{\mathtt{v} := \mathtt{w}\} = [\mathtt{v} =_{\text{Name}} \mathtt{v}] = \text{true}$$

Essentially, this is what happens in the π-calculus under late and early bisimulation: the equality test used is name equivalence, but substitution can alter the names tested. As a result, this bisimulation is not a congruence under substitution, and therefore not under input prefix.

3.2 Open Bisimulation

The various bisimulations for the π-calculus all vary in the way variable instantiations are handled. Some differences between the several proposed bisimulations can be found already in systems without a binding mechanism. Here we discuss *open* bisimulation, as proposed by [San93], which is the finest (in the sense that it equates the smallest number of processes). In order to compare our system with the one defined in [San93] we restrict our booleans to the following syntax:

$$B ::= \text{true} \mid [x =_{\text{Var}} y] \mid B \wedge B$$

A formula into this smaller system will be called a restricted boolean formula. Observe that a substitution can turn a restricted boolean into true , but not into false.

DEFINITION 3.3 A relation $\mathcal{S}$ is *closed under a substitution* σ if $P\mathcal{S}Q$ implies $P\sigma\mathcal{S}Q\sigma$.

DEFINITION 3.4 A relation $\mathcal{S}$ is a *ground simulation* if $P\mathcal{S}Q$ implies:

- whenever $P \xrightarrow{\alpha} P'$ then Q' exists s.t. $Q \xrightarrow{\alpha} Q'$ and $P'\mathcal{S}Q'$.

DEFINITION 3.5 (Sangiorgi [San93]) A relation $\mathcal{S}$ on processes is an *open* simulation if $P\mathcal{S}Q$ implies, for every σ:

- Whenever $P\sigma \xrightarrow{\alpha} P'$, then Q' exists s.t. $Q\sigma \xrightarrow{\alpha} Q'$ and $P'\mathcal{S}Q'$.

Two processes P and Q are open bisimilar, written $P \sim Q$, if $P\mathcal{S}Q$ for some open bisimulation $\mathcal{S}$.

PROPOSITION 3.6 (Sangiorgi [San93]) Supose that a relation S

1. is a ground bisimulation
2. is closed under all substitutions

Then, S is an open bisimulation.

PROPOSITION 3.7 (Sangiorgi [San93]) The relation $\sim$ is the largest ground bisimulation which is closed under all substitutions.

In the original presentation of open bisimulation two different ways to define transition relations were introduced: One equivalent to our unconditional transition relation, and other similar to our conditional one, which Sangiorgi calls "efficient characterization". The bisimulation defined for this last system differs from ours in the fact that it still uses some form of substitution.

DEFINITION 3.8 Let β be a restricted boolean formula. There is an obvious way to associate an equivalence relation on names R_β to it. We define then a special substitution σ_β which sends every name to a chosen representative in its equivalence class.

In the previous definition we can see the price one has to pay to have general booleans instead of matching sequences: we cannot define such a canonical substitution in our system, since a boolean formula does not necessarily define an equivalence relation on names.

DEFINITION 3.9 A relation S on processes is a $\asymp$-simulation if for any pair of processes P, Q it holds that PSQ implies

- whenever $P \xrightarrow{(\beta,a)} P'$ then there exist γ, b, Q' such that $Q \xrightarrow{(\gamma,b)} Q'$ and
  1. $\beta \Rightarrow \gamma$,
  2. $a\sigma_\beta = b\sigma_\beta$,
  3. $P'\sigma_\beta SQ'\sigma_\beta$.
- whenever $P \xrightarrow{(\beta,a)} \surd$ then there exist γ, b such that $Q \xrightarrow{(\gamma,b)} \surd$ and
  1. $\beta \Rightarrow \gamma$,

2. $a\sigma_\beta = b\sigma_\beta$,

A relation S is a $\asymp$-bisimulation if both S and S^{-1} are $\asymp$-simulations

The following lemma is needed for the proof of proposition 3.12

LEMMA 3.10 For $\beta, \beta' \neq$ false,

1. if $P \xrightarrow{(\beta,a)} Q$ then for any substitution σ it holds that $P\sigma \xrightarrow{(\beta',a')} Q'$ with $\beta' \Leftrightarrow \beta\sigma, a' = a\sigma$ and $Q' = Q\sigma$.

2. if for a substitution σ, $P\sigma \xrightarrow{(\beta',a')} Q'$, then $P \xrightarrow{(\beta,a)} Q$ with $\beta' \Leftrightarrow \beta\sigma, a' = a\sigma$ and $Q' = Q\sigma$.

COROLLARY 3.11 $P\sigma_\beta \xrightarrow{(\gamma\sigma_\beta, a\sigma_\beta)} Q\sigma_\beta$ if and only if $\beta :\rightarrow P \xrightarrow{(\gamma\wedge\beta,a)} Q$

PROPOSITION 3.12 $P \asymp Q$ if and only if $P \leftrightarrows Q$

PROOF Assume first that $p \asymp q$. We will construct a conditional bisimulation relating p and q.

$$S = \{(\beta :\rightarrow P, \beta :\rightarrow Q)) | P\sigma_\beta \asymp Q\sigma_\beta\}$$

We leave the proof that S is indeed a conditional bisimualtion to the reader. The fact that $\asymp$ is closed under substitution is needed in the proof.

For the other implication, we take a conditional bisimulation S and construct a $\asymp$-bisimulation as follows

$$\bar{S} = \{(P\sigma_\beta, Q\sigma_\beta) | \beta :\rightarrow P S \beta :\rightarrow Q\}$$

REMARK 3.13 As a consequence of the proof of the previous proposition we have the fact that for any condition β it holds that

$$P\sigma_\beta \asymp Q\sigma_\beta \Longleftrightarrow \beta :\rightarrow P \leftrightarrows \beta :\rightarrow Q$$

The following proposition was proved in [San93] for a system slightly different than ours which also had bound variables and more operators, like parallel composition.

PROPOSITION 3.14 With restricted booleans open bisimulation based on the unconditional inference rules for BCPA coincides with conditional bisimulation over BCPA.

PROOF See Sangiorgi [San93]: our conditional bisimulation corresponds to his "efficient characterization".

3.3 Distinctions

Distinctions were proposed in [MPW92]. They form a way of re-introducing the difference between names and variables into the π-calculus.

DEFINITION 3.15 A *distinction* is a finite symmetric and irreflexive relation on variables.

DEFINITION 3.16 We abbreviate distinctions in the following way: A set S of variables is considered an abbreviation of the distinction $(S \times S)\backslash\{(v,v) \mid v \in S\}$

The idea behind distinctions is that they express that two different names should be kept different. In our framework this can be expressed simply as the negation of an equality test. It is important that we use the predicate $[\_ =_{\mathrm{Var}} \_]$ and not $[\_ =_{\mathrm{Name}} \_]$. We first introduce the theory of distinctions presented in [San93] and then show that we can embed it in our framework without adding any extra machinery.

DEFINITION 3.17 A substitution σ *respects* a distinction D if and only if for any pair of names $(x,y) \in \mathcal{D}$ it holds that $\sigma(x) \neq \sigma(y)$. A condition β (only with equality tests and conjunctions) respects a distinction D if for every pair $(x,y) \in \mathcal{D}$ the implication $\beta \Rightarrow [x =_{\mathrm{Var}} y]$ is not true. (note that it is not necessarily equal to false).

DEFINITION 3.18 (Sangiorgi [San93]) A family of relations $\{S_{\mathcal{D}}\}_{\mathcal{D}}$ indexed by distinctions is an indexed open simulation if for all D it holds that $PS_{\mathcal{D}}Q$ implies

- whenever $P \xrightarrow{(\beta,a)} P'$, β respects D, then there exist γ, b, Q' such that $Q \xrightarrow{(\gamma,b)} Q'$ and
  1. $\beta \Rightarrow \gamma$
  2. $a\sigma_\beta = b\sigma_\beta$
  3. $P'\sigma_\beta S_{\mathcal{D}\sigma_\beta} Q'\sigma_\beta$
- whenever $P \xrightarrow{(\beta,a)} \surd$, β respects D, then there exist γ, b such that $Q \xrightarrow{(\gamma,b)} \surd$ and
  1. $\beta \Rightarrow \gamma$
  2. $a\sigma_\beta = b\sigma_\beta$

A family of relations $\{S_{\mathcal{D}}\}_{\mathcal{D}}$ indexed by distinctions is an indexed open bisimulation if both $\{S_{\mathcal{D}}\}_{\mathcal{D}}$ and its inverse $\{S_{\mathcal{D}}^{-1}\}_{\mathcal{D}}$ are indexed open simulations. We write $x \asymp_{\mathcal{D}} y$ if there is an indexed open bisimulation $\{S_{\mathcal{D}}\}_{\mathcal{D}}$ with $xS_{\mathcal{D}}y$.

DEFINITION 3.19 Given a distinction D we associate a canonical formula $\phi_{\mathcal{D}}$ to it as follows:

$$\phi_{\mathcal{D}} = \bigwedge_{(x,y)\in\mathcal{D}} \neg[x =_{\mathrm{Var}} y]$$

PROPOSITION 3.20 A general condition β (now it can contain negation and disjunction) respects a distinction D if and only if $\phi_{\mathcal{D}} \wedge \beta$ is not false.

PROOF Immediate, since $\beta \Rightarrow [x =_{\mathrm{Var}} y]$ is not true is equivalent to say that $\beta \wedge \neg[x =_{\mathrm{Var}} y]$ is not false, and since this holds for any pair $(x, y) \in \mathcal{D}$, the result follows.

The following technical proposition will be needed in lemma 3.22.

PROPOSITION 3.21 Let β be a restricted formula and D a distinction, then

$$\phi_{\mathcal{D}} \wedge \beta \Leftrightarrow \phi_{\mathcal{D}\sigma_\beta} \wedge \beta$$

LEMMA 3.22 Given a distinction D, $P \asymp_{\mathcal{D}} Q$ if and only if $\phi_{\mathcal{D}} :\rightarrow P \leftrightarrows \phi_{\mathcal{D}} :\rightarrow Q$.

PROOF

Assume first that $S_{\mathcal{D}} : P \asymp_{\mathcal{D}} Q$. We will construct a conditional bisimulation relating $\phi_{\mathcal{D}} :\rightarrow P$ and $\phi_{\mathcal{D}} :\rightarrow Q$.

$$S = \{(\beta \wedge \phi_{\mathcal{D}} :\rightarrow P, \beta \wedge \phi_{\mathcal{D}} :\rightarrow Q) | \exists \beta . P\sigma_\beta S_{\mathcal{D}} Q\sigma_\beta\}$$

We leave the proof that S is indeed a conditional bisimualtion to the reader.

For the other implication, we take a conditional bisimulation S and construct an indexed bisimulation as follows

$$S_{\mathcal{D}} = \{(P\sigma_\beta, Q\sigma_\beta) | (\phi_{\mathcal{D}} \wedge \beta :\rightarrow P) S (\phi_{\mathcal{D}} \wedge \beta :\rightarrow Q)\}$$

4 Symbolic bisimulation

In the previous section we have seen that our notion of conditional bisimulation agrees with the open bisimulation of [San93]. However, our framework is different since we do not have bound variables but on the other hand our language for booleans is richer. In this section we will show that the difference between open bisimulation and, on the other hand, early/late congruence appears already in our simplified systems. Moreover, this difference is represented by the presence of the following axiom:

SPECIFICATION 4.1

$$\beta \vee \gamma :\rightarrow x \quad = \quad \beta :\rightarrow x \, + \, \gamma :\rightarrow x \quad [\mathrm{C8}]$$

DEFINITION 4.2 Let $T = \langle S, L, \longrightarrow, \longrightarrow \surd, s_0 \rangle$ be conditional transition system. A relation $\mathcal{R} \subseteq S \times S$ is a *symbolic simulation* if $u\mathcal{R}v$ implies

- Whenever $u \xrightarrow{(\beta,a)} u'$, with $\beta \neq$ false, there exist a decomposition $\beta = \bigvee_1^n(\gamma_1, \ldots \gamma_n)$, and $\gamma'_1, \ldots \gamma'_n, v'_1, \ldots, v'_n$, such that:

  1. $v \xrightarrow{(\gamma,b_i)} v'$, with
  2. $\gamma_i \Rightarrow \gamma'_i$,

3. $\gamma_i :\rightarrow a = \gamma_i :\rightarrow b$, and
4. $(\gamma_i :\rightarrow u')\mathcal{R}(\gamma_i :\rightarrow v_i')$.

- Whenever $u \xrightarrow{(\beta,a)} \surd$, with $\beta \neq$ false, there exist a decomposition $\beta = \bigvee_1^n(\gamma_1, \ldots \gamma_n)$, and $\gamma_1', \ldots \gamma_n'$, such that:

1. $v \xrightarrow{(\gamma,b_i)} v'$, with
2. $\gamma_i \Rightarrow \gamma_i'$,
3. $\gamma_i :\rightarrow a = \gamma_i :\rightarrow b$, and

A relation $\mathcal{R}$ is a *symbolic bisimulation* if both $\mathcal{R}$ and its inverse $\mathcal{R}^{-1}$ are simulations. We write $x \leftrightarrow_s y$ if there is a bisimulation $\mathcal{R}$ with $x\mathcal{R}y$.

REMARK 4.3 It is immediate that a conditional bisimulation is a symbolic bisimulation as well.

REMARK 4.4 In the presence of C8 some of the axioms of BCPA are derivable:

C7 is derivable as follows:

$$x + \beta :\rightarrow x \overset{[GC1]}{=} \text{true} :\rightarrow x + \beta :\rightarrow x \overset{[C8]}{=} (\text{true} \vee \beta) :\rightarrow x = \text{true} :\rightarrow x = x$$

C5 is derivable as follows:

$$\begin{array}{rcl}
(\beta :\rightarrow x) \cdot y & \overset{[CG1]}{=} & \text{true} :\rightarrow ((\beta :\rightarrow x) \cdot y) \\
& = & (\beta \vee \neg\beta) :\rightarrow ((\beta :\rightarrow x) \cdot y) \\
& \overset{[C8]}{=} & \beta :\rightarrow (\beta :\rightarrow x) \cdot y + \neg\beta :\rightarrow (\beta :\rightarrow x) \cdot y \\
& \overset{[C6]}{=} & (\beta :\rightarrow (\beta :\rightarrow x)) \cdot \beta :\rightarrow y + (\neg\beta :\rightarrow (\beta :\rightarrow x)) \cdot \neg\beta :\rightarrow y \\
& \overset{[C4]}{=} & (\beta \wedge \beta :\rightarrow x) \cdot \beta :\rightarrow y + (\neg\beta \wedge \beta :\rightarrow x) \cdot \neg\beta :\rightarrow y \\
& = & (\beta :\rightarrow x) \cdot \beta :\rightarrow y + (\text{false} :\rightarrow x) \cdot \neg\beta :\rightarrow y \\
& \overset{[C6][GC2]}{=} & \beta :\rightarrow xy + \delta \\
& = & \beta :\rightarrow xy
\end{array}$$

PROPOSITION 4.5 Symbolic bisimulation equivalence for BCPA is a congruence.

PROPOSITION 4.6 (Soundness:) If $E(\text{BCPA}) + \text{C8} \vdash x = y$ then $x \leftrightarrow_s y$.

PROOF We only need to check axiom [C8], since a conditional bisimulation is also a symbolic bisimulation. It is simple to show that for any process x and conditions β and γ the relation S defined as follows:

$$\mathcal{S} = Id \cup \{(\beta \vee \gamma :\rightarrow x, \beta :\rightarrow x + \gamma :\rightarrow X)\}$$

is indeed a symbolic bisimulation.

PROPOSITION 4.7 (Completeness:) If $x \leftrightarrows_s y$ then $E(\text{BCPA}) \vdash x = y$.

PROOF

We use the same technique as for proposition 2.24. Note that lemma 2.23 can also be used here since it only depends on the definition of the transition relation, not on the equivalence.

The required completeness result is a consequence of the following property that we will prove by induction on $n(x) + n(y)$:

$$x + y \leftrightarrows_s y \Longrightarrow E(\text{BCPA+C8}) \vdash x + y = y$$

We consider only the most difficult case when

$$x = \beta :\rightarrow ax'$$

by definition of the action rules we know that

$$x + y \xrightarrow{(\beta,a)} x'$$

and then, since by hypothesis $x + y \leftrightarrows_s y$ we can conclude than

$$y \xrightarrow{(\gamma_i',b_i)} y_i'$$

where

1. $\beta = \bigvee \gamma_i$
2. $\gamma_i \Rightarrow \gamma_i'$,
3. $\gamma_i :\rightarrow a = \gamma_i :\rightarrow b_i$,
4. $\gamma_i :\rightarrow x' \leftrightarrows_s \gamma_i :\rightarrow y_i'$

Since $\leftrightarrows_s$ is a congruence we know that

$$\gamma_i :\rightarrow x' + \gamma_i :\rightarrow y_i' \leftrightarrows_s \gamma_i :\rightarrow y_i'$$

and

$$\gamma_i :\rightarrow x' + \gamma_i :\rightarrow y_i' \leftrightarrows_s \gamma_i :\rightarrow x'$$

It follows then by induction hypothesis that

$$\gamma_i :\rightarrow x' = \gamma_i :\rightarrow y_i'$$

moreover, from the second condition of the definition of bisimulation

$$(\gamma_i :\rightarrow a) \cdot \gamma_i :\rightarrow x' = (\gamma_i :\rightarrow b) \cdot \gamma_i :\rightarrow y_i'$$

or equivalently (by axiom C6)

$$\gamma_i :\rightarrow ax' = \gamma_i :\rightarrow by'_i$$

Hence, we can prove that

$$
\begin{aligned}
y &\overset{(2.23)}{=} y + \sum_{i<n} \gamma'_i :\rightarrow b_i y_i \\
&\overset{[C6]}{=} y + \sum_{i<n} \gamma'_i :\rightarrow b_i y_i + \sum_{i<n} \gamma_i :\rightarrow \gamma'_i :\rightarrow b_i y_i \\
&\overset{[C4]}{=} y + \sum_{i<n} \gamma_i \wedge \gamma'_i :\rightarrow b_i y_i \\
&\overset{\gamma_i \Rightarrow \gamma'_i}{=} y + \sum_{i<n} \gamma_i :\rightarrow b_i y_i \\
&= y + \sum_{i<n} \gamma_i :\rightarrow ax' \\
&\overset{[C8]}{=} y + (\bigvee_{i<n} \gamma_i) :\rightarrow ax' \\
&= y + x
\end{aligned}
$$

PROPOSITION 4.8 BPA is a Reduced Model Specification of BCPA: The initial algebra of BPA is a subalgebra of the initial algebra of BCPA.

PROPOSITION 4.9 BCPA + C8 is a conservative extension of BPA.

EXAMPLE 4.10 Consider the following three processes:

$$P = \tau \cdot \tau + \tau$$

$$Q = \tau \cdot \tau + \tau + \tau \cdot [x =_{\mathrm{Var}} y] :\rightarrow \tau$$

$$R = \tau \cdot \tau + \tau \cdot \delta$$

They are not open bisimilar. The process Q is symbolic bisimilar to R but not to P. This difference with a similar example presented in [San93] is due to the presence of both successful and unsuccessful termination in our algebra and the identification of the second with a process with a false condition.

5 Concluding Remarks

A simple process algebra was introduced, in which conditions play an important role from the start. This approach simplifies the completeness proofs and allows a simple comparison between (a generalization of) open bisimulation and late and early bisimulations. We showed that the main difference between both relies on the way in which the conditions are evaluated, whereas the difference between late and early bisimulations can only be expressed by using different ways to instantiate the variables.

6 Further Work

The aim of defining basic conditional process algebra is to use it as a starting point to build a ACP-style algebra of mobile processes. The next step will be the introduction of binding mechanisms for names, restriction and communication.

It may be possible to combine conditional bisimulation with an early scheme for the instantiation of variables, by only changing the action relations for input prefix and communication as in [MPW91]. This can combine the good property of open bisimulation of being a congruence for all the operators with the possibility to eliminate bound input in terms of free input as in [BB94].

Given the logic-oriented presentation of the different semantics it seems natural also to look for modal logics that characterize conditional and symbolic bisimulation in our framework.

7 Related Work

Many works on ACP-style algebras introduced some form of conditions on processes, for example [GP94]. However, in all these works, axiom [C6] was missing. This axiom means intuitively that the knowledge about names can only increase with time. The conditional bisimulation introduced here generalizes open bisimulation from [San93] to a framework where also negation is present in a conservative way. The presence of negation (or at least inequations) allows us to introduce (a generalization of) distinctions without any extra machinery. The symbolic bisimulation is very similar to the one introduced in [HL95], but it is presented here in a less abstract way.

References

[BB92] J.C.M. Baeten and J.A. Bergstra. Process algebra with signals and conditions. In M. Broy, editor, *Programming and Mathematical Method, Marktoberdorf 1990*, number F 88 in NATO ASI Series, pages 273–323. Springer-Verlag, 1992.

[BB94] J.C.M. Baeten and J.A. Bergstra. On sequential composition, action prefixes and process prefix. *Formal Aspects of Computing*, 6:250–268, 1994.

[BB95] J.C.M. Baeten and J.A. Bergstra. Process algebra with propositional signals. In A. Ponse, C. Verhoef, and B. van Vlijmen, editors, *Algebra of Communicating Processes, ACP'95*, 1995.

[BV95] J.C.M. Baeten and C. Verhoef. Concrete process algebra. In S. Abramsky, D.M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science, Volume 4*. Oxford University Press, 1995.

[GP94] J.F. Groote and A. Ponse. Process algebra with guards: Combining Hoare logic and process algebra. *Formal Aspects of Computing*, 6(2):115–164, 1994.

[HL95] M. Hennesy and H. Lin. Symbolic bisimulation. *Theoretical Computer Science*, 138:353–389, 1995.

[MPW91] R. Milner, J. Parrow, and D. Walker. Modal logics for mobile processes. In J.C.M. Baeten and J.F. Groote, editors, *Proceedings CONCUR'91*, volume 527 of *LNCS*, pages 45–60. Springer-Verlag, 1991.

[MPW92] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes. *Information and Computation*, 100:1–77, 1992.

[PS93] J. Parrow and D. Sangiorgi. Algebraic theories for name-passing calculi. In J.W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *REX – A Decade of Concurrency: Reflections and Perspectives*, volume 803 of *LNCS*. Springer-Verlag, 1993.

[San93] D. Sangiorgi. A theory of bisimimulation for the π-calculus. In E. Best, editor, *Proceedings of CONCUR'93*, volume 715 of *LNCS*, pages 127–142. Springer-Verlag, 1993.

Computing Science Reports

In this series appeared:

Department of Mathematics and Computing Science
Eindhoven University of Technology

93/01	R. van Geldrop	Deriving the Aho-Corasick algorithms: a case study into the synergy of programming methods, p. 36.
93/02	T. Verhoeff	A continuous version of the Prisoner's Dilemma, p. 17
93/03	T. Verhoeff	Quicksort for linked lists, p. 8.
93/04	E.H.L. Aarts J.H.M. Korst P.J. Zwietering	Deterministic and randomized local search, p. 78.
93/05	J.C.M. Baeten C. Verhoef	A congruence theorem for structured operational semantics with predicates, p. 18.
93/06	J.P. Veltkamp	On the unavoidability of metastable behaviour, p. 29
93/07	P.D. Moerland	Exercises in Multiprogramming, p. 97
93/08	J. Verhoosel	A Formal Deterministic Scheduling Model for Hard Real-Time Executions in DEDOS, p. 32.
93/09	K.M. van Hee	Systems Engineering: a Formal Approach Part I: System Concepts, p. 72.
93/10	K.M. van Hee	Systems Engineering: a Formal Approach Part II: Frameworks, p. 44.
93/11	K.M. van Hee	Systems Engineering: a Formal Approach Part III: Modeling Methods, p. 101.
93/12	K.M. van Hee	Systems Engineering: a Formal Approach Part IV: Analysis Methods, p. 63.
93/13	K.M. van Hee	Systems Engineering: a Formal Approach Part V: Specification Language, p. 89.
93/14	J.C.M. Baeten J.A. Bergstra	On Sequential Composition, Action Prefixes and Process Prefix, p. 21.
93/15	J.C.M. Baeten J.A. Bergstra R.N. Bol	A Real-Time Process Logic, p. 31.
93/16	H. Schepers J. Hooman	A Trace-Based Compositional Proof Theory for Fault Tolerant Distributed Systems, p. 27
93/17	D. Alstein P. van der Stok	Hard Real-Time Reliable Multicast in the DEDOS system, p. 19.
93/18	C. Verhoef	A congruence theorem for structured operational semantics with predicates and negative premises, p. 22.
93/19	G-J. Houben	The Design of an Online Help Facility for ExSpect, p.21.
93/20	F.S. de Boer	A Process Algebra of Concurrent Constraint Programming, p. 15.
93/21	M. Codish D. Dams G. Filé M. Bruynooghe	Freeness Analysis for Logic Programs - And Correctness, p. 24
93/22	E. Poll	A Typechecker for Bijective Pure Type Systems, p. 28.
93/23	E. de Kogel	Relational Algebra and Equational Proofs, p. 23.
93/24	E. Poll and Paula Severi	Pure Type Systems with Definitions, p. 38.
93/25	H. Schepers and R. Gerth	A Compositional Proof Theory for Fault Tolerant Real-Time Distributed Systems, p. 31.
93/26	W.M.P. van der Aalst	Multi-dimensional Petri nets, p. 25.
93/27	T. Kloks and D. Kratsch	Finding all minimal separators of a graph, p. 11.
93/28	F. Kamareddine and R. Nederpelt	A Semantics for a fine λ-calculus with de Bruijn indices, p. 49.
93/29	R. Post and P. De Bra	GOLD, a Graph Oriented Language for Databases, p. 42.
93/30	J. Deogun T. Kloks D. Kratsch H. Müller	On Vertex Ranking for Permutation and Other Graphs, p. 11.

93/31	W. Körver	Derivation of delay insensitive and speed independent CMOS circuits, using directed commands and production rule sets, p. 40.
93/32	H. ten Eikelder and H. van Geldrop	On the Correctness of some Algorithms to generate Finite Automata for Regular Expressions, p. 17.
93/33	L. Loyens and J. Moonen	ILIAS, a sequential language for parallel matrix computations, p. 20.
93/34	J.C.M. Baeten and J.A. Bergstra	Real Time Process Algebra with Infinitesimals, p.39.
93/35	W. Ferrer and P. Severi	Abstract Reduction and Topology, p. 28.
93/36	J.C.M. Baeten and J.A. Bergstra	Non Interleaving Process Algebra, p. 17.
93/37	J. Brunekreef J-P. Katoen R. Koymans S. Mauw	Design and Analysis of Dynamic Leader Election Protocols in Broadcast Networks, p. 73.
93/38	C. Verhoef	A general conservative extension theorem in process algebra, p. 17.
93/39	W.P.M. Nuijten E.H.L. Aarts D.A.A. van Erp Taalman Kip K.M. van Hee	Job Shop Scheduling by Constraint Satisfaction, p. 22.
93/40	P.D.V. van der Stok M.M.M.P.J. Claessen D. Alstein	A Hierarchical Membership Protocol for Synchronous Distributed Systems, p. 43.
93/41	A. Bijlsma	Temporal operators viewed as predicate transformers, p. 11.
93/42	P.M.P. Rambags	Automatic Verification of Regular Protocols in P/T Nets, p. 23.
93/43	B.W. Watson	A taxomomy of finite automata construction algorithms, p. 87.
93/44	B.W. Watson	A taxonomy of finite automata minimization algorithms, p. 23.
93/45	E.J. Luit J.M.M. Martin	A precise clock synchronization protocol,p.
93/46	T. Kloks D. Kratsch J. Spinrad	Treewidth and Patwidth of Cocomparability graphs of Bounded Dimension, p. 14.
93/47	W. v.d. Aalst P. De Bra G.J. Houben Y. Komatzky	Browsing Semantics in the "Tower" Model, p. 19.
93/48	R. Gerth	Verifying Sequentially Consistent Memory using Interface Refinement, p. 20.
94/01	P. America M. van der Kammen R.P. Nederpelt O.S. van Roosmalen H.C.M. de Swart	The object-oriented paradigm, p. 28.
94/02	F. Kamareddine R.P. Nederpelt	Canonical typing and Π-conversion, p. 51.
94/03	L.B. Hartman K.M. van Hee	Application of Marcov Decision Processe to Search Problems, p. 21.
94/04	J.C.M. Baeten J.A. Bergstra	Graph Isomorphism Models for Non Interleaving Process Algebra, p. 18.
94/05	P. Zhou J. Hooman	Formal Specification and Compositional Verification of an Atomic Broadcast Protocol, p. 22.
94/06	T. Basten T. Kunz J. Black M. Coffin D. Taylor	Time and the Order of Abstract Events in Distributed Computations, p. 29.
94/07	K.R. Apt R. Bol	Logic Programming and Negation: A Survey, p. 62.
94/08	O.S. van Roosmalen	A Hierarchical Diagrammatic Representation of Class Structure, p. 22.
94/09	J.C.M. Baeten J.A. Bergstra	Process Algebra with Partial Choice, p. 16.

94/10	T. verhoeff	The testing Paradigm Applied to Network Structure. p. 31.
94/11	J. Peleska C. Huizing C. Petersohn	A Comparison of Ward & Mellor's Transformation Schema with State- & Activitycharts, p. 30.
94/12	T. Kloks D. Kratsch H. Müller	Dominoes, p. 14.
94/13	R. Seljée	A New Method for Integrity Constraint checking in Deductive Databases, p. 34.
94/14	W. Peremans	Ups and Downs of Type Theory, p. 9.
94/15	R.J.M. Vaessens E.H.L. Aarts J.K. Lenstra	Job Shop Scheduling by Local Search, p. 21.
94/16	R.C. Backhouse H. Doornbos	Mathematical Induction Made Calculational, p. 36.
94/17	S. Mauw M.A. Reniers	An Algebraic Semantics of Basic Message Sequence Charts, p. 9.
94/18	F. Kamareddine R. Nederpelt	Refining Reduction in the Lambda Calculus, p. 15.
94/19	B.W. Watson	The performance of single-keyword and multiple-keyword pattern matching algorithms, p. 46.
94/20	R. Bloo F. Kamareddine R. Nederpelt	Beyond β-Reduction in Church's $\lambda\!\rightarrow$, p. 22.
94/21	B.W. Watson	An introduction to the Fire engine: A C++ toolkit for Finite automata and Regular Expressions.
94/22	B.W. Watson	The design and implementation of the FIRE engine: A C++ toolkit for Finite automata and regular Expressions.
94/23	S. Mauw and M.A. Reniers	An algebraic semantics of Message Sequence Charts, p. 43.
94/24	D. Dams O. Grumberg R. Gerth	Abstract Interpretation of Reactive Systems: Abstractions Preserving $\forall$CTL*, $\exists$CTL* and CTL*, p. 28.
94/25	T. Kloks	$K_{1,3}$-free and W_4-free graphs, p. 10.
94/26	R.R. Hoogerwoord	On the foundations of functional programming: a programmer's point of view, p. 54.
94/27	S. Mauw and H. Mulder	Regularity of BPA-Systems is Decidable, p. 14.
94/28	C.W.A.M. van Overveld M. Verhoeven	Stars or Stripes: a comparative study of finite and transfinite techniques for surface modelling, p. 20.
94/29	J. Hooman	Correctness of Real Time Systems by Construction, p. 22.
94/30	J.C.M. Baeten J.A. Bergstra Gh. Ştefanescu	Process Algebra with Feedback, p. 22.
94/31	B.W. Watson R.E. Watson	A Boyer-Moore type algorithm for regular expression pattern matching, p. 22.
94/32	J.J. Vereijken	Fischer's Protocol in Timed Process Algebra, p. 38.
94/33	T. Laan	A formalization of the Ramified Type Theory, p.40.
94/34	R. Bloo F. Kamareddine R. Nederpelt	The Barendregt Cube with Definitions and Generalised Reduction, p. 37.
94/35	J.C.M. Baeten S. Mauw	Delayed choice: an operator for joining Message Sequence Charts, p. 15.
94/36	F. Kamareddine R. Nederpelt	Canonical typing and Π-conversion in the Barendregt Cube, p. 19.
94/37	T. Basten R. Bol M. Voorhoeve	Simulating and Analyzing Railway Interlockings in ExSpect, p. 30.
94/38	A. Bijlsma C.S. Scholten	Point-free substitution, p. 10.
94/39	A. Blokhuis T. Kloks	On the equivalence covering number of splitgraphs, p. 4.

94/40	D. Alstein	Distributed Consensus and Hard Real-Time Systems, p. 34.
94/41	T. Kloks D. Kratsch	Computing a perfect edge without vertex elimination ordering of a chordal bipartite graph, p. 6.
94/42	J. Engelfriet J.J. Vereijken	Concatenation of Graphs, p. 7.
94/43	R.C. Backhouse M. Bijsterveld	Category Theory as Coherently Constructive Lattice Theory: An Illustration, p. 35.
94/44	E. Brinksma, J. Davies R. Gerth, S. Graf W. Janssen, B. Jonsson S. Katz, G. Lowe M. Poel, A. Pnueli C. Rump, J. Zwiers	Verifying Sequentially Consistent Memory, p. 160
94/45	G.J. Houben	Tutorial voor de ExSpect-bibliotheek voor "Administratieve Logistiek", p. 43.
94/46	R. Bloo F. Kamareddine R. Nederpelt	The λ-cube with classes of terms modulo conversion, p. 16.
94/47	R. Bloo F. Kamareddine R. Nederpelt	On Π-conversion in Type Theory, p. 12.
94/48	Mathematics of Program Construction Group	Fixed-Point Calculus, p. 11.
94/49	J.C.M. Baeten J.A. Bergstra	Process Algebra with Propositional Signals, p. 25.
94/50	H. Geuvers	A short and flexible proof of Strong Normalazation for the Calculus of Constructions, p. 27.
94/51	T. Kloks D. Kratsch H. Müller	Listing simplicial vertices and recognizing diamond-free graphs, p. 4.
94/52	W. Penczek R. Kuiper	Traces and Logic, p. 81
94/53	R. Gerth R. Kuiper D. Peled W. Penczek	A Partial Order Approach to Branching Time Logic Model Checking, p. 20.
95/01	J.J. Lukkien	The Construction of a small CommunicationLibrary, p.16.
95/02	M. Bezem R. Bol J.F. Groote	Formalizing Process Algebraic Verifications in the Calculus of Constructions, p.49.
95/03	J.C.M. Baeten C. Verhoef	Concrete process algebra, p. 134.
95/04	J. Hidders	An Isotopic Invariant for Planar Drawings of Connected Planar Graphs, p. 9.
95/05	P. Severi	A Type Inference Algorithm for Pure Type Systems, p.20.
95/06	T.W.M. Vossen M.G.A. Verhoeven H.M.M. ten Eikelder E.H.L. Aarts	A Quantitative Analysis of Iterated Local Search, p.23.
95/07	G.A.M. de Bruyn O.S. van Roosmalen	Drawing Execution Graphs by Parsing, p. 10.
95/08	R. Bloo	Preservation of Strong Normalisation for Explicit Substitution, p. 12.
95/09	J.C.M. Baeten J.A. Bergstra	Discrete Time Process Algebra, p. 20
95/10	R.C. Backhouse R. Verhoeven O. Weber	Math∫pad: A System for On-Line Prepararation of Mathematical Documents, p. 15
95/11	R. Seljée	Deductive Database Systems and integrity constraint checking, p. 36.
95/12	S. Mauw and M. Reniers	Empty Interworkings and Refinement

		Semantics of Interworkings Revised, p. 19.
95/13	B.W. Watson and G. Zwaan	A taxonomy of sublinear multiple keyword pattern matching algorithms, p. 26.
95/14	A. Ponse, C. Verhoef, S.F.M. Vlijmen (eds.)	De proceedings: ACP'95, p.
95/15	P. Niebert and W. Penczek	On the Connection of Partial Order Logics and Partial Order Reduction Methods, p. 12.
95/16	D. Dams, O. Grumberg, R. Gerth	Abstract Interpretation of Reactive Systems: Preservation of CTL*, p. 27.
95/17	S. Mauw and E.A. van der Meulen	Specification of tools for Message Sequence Charts, p. 36.
95/18	F. Kamareddine and T. Laan	A Reflection on Russell's Ramified Types and Kripke's Hierarchy of Truths, p. 14.
95/19	J.C.M. Baeten and J.A. Bergstra	Discrete Time Process Algebra with Abstraction, p. 15.
95/20	F. van Raamsdonk and P. Severi	On Normalisation, p. 33.
95/21	A. van Deursen	Axiomatizing Early and Late Input by Variable Elimination, p. 44.
95/22	B. Arnold, A. v. Deursen, M. Res	An Algebraic Specification of a Language for Describing Financial Products, p. 11.
95/23	W.M.P. van der Aalst	Petri net based scheduling, p. 20.
95/24	F.P.M. Dignum, W.P.M. Nuijten, L.M.A. Janssen	Solving a Time Tabling Problem by Constraint Satisfaction, p. 14.
95/25	L. Feijs	Synchronous Sequence Charts In Action, p. 36.
95/26	W.M.P. van der Aalst	A Class of Petri nets for modeling and analyzing business processes, p. 24.
95/27	P.D.V. van der Stok, J. van der Wal	Proceedings of the Real-Time Database Workshop, p. 106.
95/28	W. Fokkink, C. Verhoef	A Conservative Look at term Deduction Systems with Variable Binding, p. 29.
95/29	H. Jurjus	On Nesting of a Nonmonotonic Conditional, p. 14
95/30	J. Hidders, C. Hoskens, J. Paredaens	The Formal Model of a Pattern Browsing Technique, p.24.
95/31	P. Kelb, D. Dams and R. Gerth	Practical Symbolic Model Checking of the full μ-calculus using Compositional Abstractions, p. 17.
95/32	W.M.P. van der Aalst	Handboek simulatie, p. 51.
95/33	J. Engelfriet and JJ. Vereijken	Context-Free Graph Grammars and Concatenation of Graphs, p. 35.
95/34	J. Zwanenburg	Record concatenation with intersection types, p. 46.
95/35	T. Basten and M. Voorhoeve	An algebraic semantics for hierarchical P/T Nets, p. 32.
96/01	M. Voorhoeve and T. Basten	Process Algebra with Autonomous Actions, p. 12.
96/02	P. de Bra and A. Aerts	Multi-User Publishing in the Web: DreSS, A Document Repository Service Station, p. 12
96/03	W.M.P. van der Aalst	Parallel Computation of Reachable Dead States in a Free-choice Petri Net, p. 26.
96/04	S. Mauw	Example specifications in phi-SDL.
96/05	T. Basten and W.M.P. v.d. Aalst	A Process-Algebraic Approach to Life-Cycle Inheritance Inheritance = Encapsulation + Abstraction, p. 15.
96/06	W.M.P. van der Aalst and T. Basten	Life-Cycle Inheritance A Petri-Net-Based Approach, p. 18.
96/07	M. Voorhoeve	Structural Petri Net Equivalence, p. 16.
96/08	A.T.M. Aerts, P.M.E. De Bra, J.T. de Munk	OODB Support for WWW Applications: Disclosing the internal structure of Hyperdocuments, p. 14.
96/09	F. Dignum, H. Weigand, E. Verharen	A Formal Specification of Deadlines using Dynamic Deontic Logic, p. 18.
96/10	R. Bloo, H. Geuvers	Explicit Substitution: on the Edge of Strong Normalisation, p. 13.
96/11	T. Laan	AUTOMATH and Pure Type Systems, p. 30.
96/12	F. Kamareddine and T. Laan	A Correspondence between Nuprl and the Ramified Theory of Types, p. 12.
96/13	T. Borghuis	Priorean Tense Logics in Modal Pure Type Systems, p. 61
96/14	S.H.J. Bos and M.A. Reniers	The I^2C-bus in Discrete-Time Process Algebra, p. 25.
96/15	M.A. Reniers and J.J. Vereijken	Completeness in Discrete-Time Process Algebra, p. 139.
96/16	P. Hoogendijk and O. de Moor	What is a data type?, p. 29.

96/17	E. Boiten and P. Hoogendijk	Nested collections and polytypism, p. 11.
96/18	P.D.V. van der Stok	Real-Time Distributed Concurrency Control Algorithms with mixed time constraints, p. 71.
96/19	M.A. Reniers	Static Semantics of Message Sequence Charts, p. 71
96/20	L. Feijs	Algebraic Specification and Simulation of Lazy Functional Programs in a concurrent Environment, p. 27.
96/21	L. Bijlsma and R. Nederpelt	Predicate calculus: concepts and misconceptions, p. 26.
96/22	M.C.A. van de Graaf and G.J. Houben	Designing Effective Workflow Management Processes, p. 22.
96/23	W.M.P. van der Aalst	Structural Characterizations of sound workflow nets, p. 22.
96/24	M. Voorhoeve and W. van der Aalst	Conservative Adaption of Workflow, p.22
96/25	M. Vaccari and R.C. Backhouse	Deriving a systolic regular language recognizer, p. 28
97/01	B. Knaack and R. Gerth	A Discretisation Method for Asynchronous Timed Systems.
97/02	J. Hooman and O. v. Roosmalen	A Programming-Language Extension for Distributed Real-Time Systems, p. 50.